

Near-Optimal ε -Kernel Construction and Related Problems

Sunil Arya*

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
arya@cse.ust.hk

Guilherme D. da Fonseca
Université Clermont Auvergne and
LIMOS
Clermont-Ferrand, France
fonseca@isima.fr

David M. Mount†
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742
mount@cs.umd.edu

Abstract

The computation of (i) ε -kernels, (ii) approximate diameter, and (iii) approximate bichromatic closest pair are fundamental problems in geometric approximation. In this paper, we describe new algorithms that offer significant improvements to their running times. In each case the input is a set of n points in \mathbb{R}^d for a constant dimension $d \geq 3$ and an approximation parameter $\varepsilon > 0$. We reduce the respective running times
(i) from $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ to $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$,
(ii) from $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ to $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$, and
(iii) from $O(n/\varepsilon^{d/3})$ to $O(n/\varepsilon^{d/4+\alpha})$,
for an arbitrarily small constant $\alpha > 0$. Result (i) is nearly optimal since the size of the output ε -kernel is $\Theta(1/\varepsilon^{(d-1)/2})$ in the worst case.

These results are all based on an efficient decomposition of a convex body using a hierarchy of Macbeath regions, and contrast to previous solutions that decompose space using quadrees and grids. By further application of these techniques, we also show that it is possible to obtain near-optimal preprocessing time for the most efficient data structures to approximately answer queries for (iv) nearest-neighbor searching, (v) directional width, and (vi) polytope membership.

1 Introduction

In this paper we present new faster algorithms to several fundamental geometric approximation problems involving point sets in d -dimensional space. In particular, we present approximation algorithms for ε -kernels, diameter, bichromatic closest pair, and the minimum bottleneck spanning tree. Our results arise from a recently developed shape-sensitive approach to approximating convex bodies, which is based on the classical concept of Macbeath regions. This approach has been applied to computing area-sensitive bounds for polytope approximation [6], polytope approximations with low combinatorial complexity [7], answering approximate polytope-membership queries [8], and approximate nearest-neighbor searching [8]. The results of [8] demonstrated the existence of data structures for these query problems but did not discuss preprocessing in detail. We complete the story by presenting efficient algorithms for building data structures for three related queries: approximate polytope membership, approximate directional width, and approximate nearest-neighbors.

*Research supported by the Research Grants Council of Hong Kong, China under project number 16200014.

†Research supported by NSF grant CCF-1618866.

Throughout, we assume that the dimension d is a constant. Our running times will often involve expressions of the form $1/\varepsilon^\alpha$. In such cases, $\alpha > 0$ is constant that can be made arbitrarily small. The approximation parameter ε is treated as an asymptotic variable that approaches 0. We assume throughout that $\varepsilon < 1$, which guarantees that $\log \frac{1}{\varepsilon} > 0$.

In Section 1.1, we present our results for ε -kernels, diameter, bichromatic closest pair, and minimum bottleneck tree. In Section 1.2, we present our results for the data structure problems. In Section 1.3, we give an overview of the techniques used.

Concurrently and independently, Timothy Chan has reported complexity bounds that are very similar to our results [20]. Remarkably, the computational techniques seem to be very different, based on Chebyshev polynomials.

1.1 Static Results

Kernel. Given a set S of n points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, an ε -coreset is an (ideally small) subset of S that approximates some measure over S (see [2] for a survey). Given a nonzero vector $v \in \mathbb{R}^d$, the *directional width* of S in direction v , $\text{width}_v(S)$ is the minimum distance between two hyperplanes that enclose S and are orthogonal to v . A *coreset for the directional width* (also known as an ε -kernel and as a *coreset for the extent measure*) is a subset $Q \subseteq S$ such that $\text{width}_v(Q) \geq (1 - \varepsilon) \text{width}_v(S)$, for all $v \in \mathbb{R}^d$. Kernels are among the most fundamental constructions in geometric approximation, playing a role similar to that of convex hulls in exact computations. Kernels have been used to obtain approximation algorithms to several problems such as diameter, minimum width, convex hull volume, minimum enclosing cylinder, minimum enclosing annulus, and minimum-width cylindrical shell [1, 2].

The concept of ε -kernels was introduced by Agarwal et al. [1]. The existence of ε -kernels with $O(1/\varepsilon^{(d-1)/2})$ points is implied in the works of Dudley [22] and Bronshteyn and Ivanov [18], and this is known to be optimal in the worst case. Agarwal et al. [1] demonstrated how to compute such a kernel in $O(n + 1/\varepsilon^{3(d-1)/2})$ time, which reduces to $O(n)$ when $n = \Omega(1/\varepsilon^{3(d-1)/2})$. While less succinct ε -kernels with $O(1/\varepsilon^{d-1})$ points can be constructed in time $O(n)$ for all n [1, 16], no linear-time algorithm is known to build an ε -kernel of optimal size. Hereafter, we use the term ε -kernel to refer exclusively to an ε -kernel of size $O(1/\varepsilon^{(d-1)/2})$.

Chan [19] showed that an ε -kernel can be constructed in $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ time, which is nearly linear when $n = \Omega(1/\varepsilon^{d-2})$. He posed the open problem of obtaining a faster algorithm. A decade later, Arya and Chan [4] showed how to build an ε -kernel in roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ time using discrete Voronoi diagrams. In this paper, we attain the following near-optimal construction time.

Theorem 1.1. *Given a set S of n points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, it is possible to construct an ε -kernel of S with $O(1/\varepsilon^{(d-1)/2})$ points in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time, where α is an arbitrarily small positive constant.*

We note that when $n = o(1/\varepsilon^{(d-1)/2})$, the input S is already an ε -kernel and therefore an $O(n)$ time algorithm is trivial. Because the worst-case output size is $O(1/\varepsilon^{(d-1)/2})$, we may assume that n is at least this large, for otherwise we can simply take S itself to be the kernel. Since $1/\varepsilon^\alpha$ dominates $\log \frac{1}{\varepsilon}$, the above running time can be expressed as $O(n/\varepsilon^\alpha)$, which is nearly linear given that α can be made arbitrarily small.

Diameter. An important application of ε -kernels is to approximate the diameter of a point set. Given n data points, the *diameter* is defined to be the maximum distance between any two data points. An ε -approximation of the diameter is a pair of points whose distance is at least $(1 - \varepsilon)$ times the exact diameter. There are multiple algorithms to approximate the diameter [1, 3, 4, 15, 19]. The fastest running times are $O((n + 1/\varepsilon^{d-2}) \log \frac{1}{\varepsilon})$ [19] and roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ [4]. The algorithm from [19] essentially computes an ε -kernel Q and then determines the maximum value of $\text{width}_v(Q)$ among a set of $k = O(1/\varepsilon^{(d-1)/2})$ directions v by brute force [1]. Discrete Voronoi diagrams [4] permit this computation in roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ time. Therefore, combining the kernel construction of Theorem 1.1 with discrete Voronoi diagrams [4], we reduce n to $O(1/\varepsilon^{(d-1)/2})$ and obtain an algorithm to ε -approximate the diameter in roughly $O(n + 1/\varepsilon^{3d/4})$ time. However, we show that it is possible to obtain a much faster algorithm, as presented in the following theorem.

Theorem 1.2. *Given a set S of n points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, it is possible to compute an ε -approximation to the diameter of S in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time.*

Bichromatic Closest Pair. In the *bichromatic closest pair* (BCP) problem, we are given n points from two sets, designated red and blue, and we want to find the closest red-blue pair. In the ε -approximate version, the goal is to find a red-blue pair of points whose distance is at most $(1 + \varepsilon)$ times the exact BCP distance. Approximations to the BCP problem were introduced in [26], and the most efficient randomized approximation algorithm runs in roughly $O(n/\varepsilon^{d/3})$ expected time [4]. We present the following result.

Theorem 1.3. *Given n red and blue points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, there is a randomized algorithm that computes an ε -approximation to the bichromatic closest pair in $O(n/\varepsilon^{d/4+\alpha})$ expected time.*

Euclidean Trees. Given a set S of n points in \mathbb{R}^d , a *Euclidean minimum spanning tree* is the spanning tree with vertex set S that minimizes the sum of the edge lengths, while a *Euclidean minimum bottleneck tree* minimizes the maximum edge length. In the approximate version we respectively approximate the sum and the maximum of the edge lengths. A minimum spanning tree is a minimum bottleneck tree (although the converse does not hold). However, an approximation to the minimum spanning tree is not necessarily an approximation to the minimum bottleneck tree. A recent approximation algorithm to the Euclidean minimum spanning tree takes roughly $O(n \log n + n/\varepsilon^2)$ time, regardless of the (constant) dimension [11]. On the other hand, the fastest algorithm to approximate the minimum bottleneck tree takes roughly $O((n \log n)/\varepsilon^{d/3})$ expected time [4]. The algorithm uses BCP to simultaneously attain an approximation to the minimum bottleneck and the minimum spanning trees. We prove the following theorem.

Theorem 1.4. *Given n points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, there is a randomized algorithm that computes a tree T that is an ε -approximation to both the Euclidean minimum bottleneck and the Euclidean minimum spanning trees in $O((n \log n)/\varepsilon^{d/4+\alpha})$ expected time.*

1.2 Data Structure Results

Polytope membership. Let P denote a convex polytope in \mathbb{R}^d , represented as the bounded intersection of n halfspaces. The *polytope membership problem* consists of preprocessing P so that it is possible to determine efficiently whether a given query point $q \in \mathbb{R}^d$ lies within P . In the ε -approximate version, we consider an expanded convex body $K \supset P$. A natural way to define this expansion would be to consider the set of points that lie within distance $\varepsilon \cdot \text{diam}(P)$ of P , thus defining a body whose Hausdorff distance from P is $\varepsilon \cdot \text{diam}(P)$. However, this definition has the shortcoming that it is not sensitive to the directional width of P . Instead, we define K as follows. For any nonzero vector $v \in \mathbb{R}^d$, consider the two supporting hyperplanes for P that are normal to v . Translate each of these hyperplanes outward by a distance of $\varepsilon \cdot \text{width}_v(P)$, and consider the closed slab-like region lying between them. Define K to be the intersection of this (infinite) set of slabs. This is clearly a stronger approximation than the Hausdorff-based definition. An ε -approximate polytope membership query (ε -APM query) returns a positive result if the query point q is inside P , a negative result if q is outside K , and may return either result otherwise.¹

We recently proposed an optimal data structure to answer approximate polytope membership queries, but efficient preprocessing remained an open problem [8]. In this paper, we present a similar data structure that not only attains optimal storage and query time, but can also be preprocessed in near-optimal time.

Theorem 1.5. *Given a convex polytope P in \mathbb{R}^d represented as the intersection of n halfspaces and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε -approximate polytope membership queries with*

$$\text{Query time: } O\left(\log \frac{1}{\varepsilon}\right) \quad \text{Space: } O\left(1/\varepsilon^{\frac{d-1}{2}}\right) \quad \text{Preprocessing: } O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}+\alpha}\right).$$

¹Our earlier works on ε -APM queries [5, 8] use the weaker Hausdorff form to define the problem, but the solutions presented there actually achieve the stronger direction-sensitive form.

Directional width. Applying the previous data structure in the dual space, we obtain a data structure for the following ε -approximate directional width problem, which is closely related to ε -kernels. Given a set S of n points in a constant dimension d and an approximation parameter $\varepsilon > 0$, the goal is to preprocess S to efficiently ε -approximate $\text{width}_v(S)$, for a nonzero query vector v . We present the following result.

Theorem 1.6. *Given a set S of n points in \mathbb{R}^d and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε -approximate directional width queries with*

$$\text{Query time: } O\left(\log^2 \frac{1}{\varepsilon}\right) \text{ Space: } O\left(1/\varepsilon^{\frac{d-1}{2}}\right) \text{ Preprocessing: } O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2} + \alpha}\right).$$

Nearest Neighbor. Let S be a set of n points in \mathbb{R}^d . Given any $q \in \mathbb{R}^d$, an ε -approximate nearest neighbor (ANN) of q is any point of S whose distance from q is at most $(1+\varepsilon)$ times the distance to q 's closest point in S . The objective is to preprocess S in order to answer such queries efficiently. Data structures for approximate nearest neighbor searching (in fixed dimensions) have been proposed by several authors, offering space-time tradeoffs (see [8] for an overview of the tradeoffs). Applying the reduction from approximate nearest neighbor to approximate polytope membership established in [5] together with Theorem 1.5, we obtain the following result, which matches the best bound [8] up to an $O(\log \frac{1}{\varepsilon})$ factor in the query time, but offers faster preprocessing time.

Theorem 1.7. *Given a set S of n points in \mathbb{R}^d , an approximation parameter $\varepsilon > 0$, and m such that $\log \frac{1}{\varepsilon} \leq m \leq 1/(\varepsilon^{d/2} \log \frac{1}{\varepsilon})$, there is a data structure that can answer Euclidean ε -approximate nearest neighbor queries with*

$$\text{Query time: } O\left(\log n + \frac{\log \frac{1}{\varepsilon}}{m \cdot \varepsilon^{\frac{d}{2}}}\right) \text{ Space: } O(nm) \text{ Preprocessing: } O\left(n \log n \log \frac{1}{\varepsilon} + \frac{nm}{\varepsilon^\alpha}\right).$$

1.3 Techniques

In contrast to previous kernel constructions, which are based on grids and the execution of Bronshteyn and Ivanov's algorithm, our construction employs a classical structure from the theory of convexity, called *Macbeath regions* [27]. Macbeath regions have found numerous uses in the theory of convex sets and the geometry of numbers (see Bárány [14] for an excellent survey). They have also been applied to several problems in the field of computational geometry. However, most previous results were either in the form of lower bounds [9, 12, 17] or focused on existential results [6, 7, 23, 30].

In [8] the authors introduced a data structure based on a hierarchy of ellipsoids based on Macbeath regions to answer approximate polytope membership queries, but the efficient computation of the hierarchy was not considered. In this paper, we show how to efficiently construct the Macbeath regions that form the basis of this hierarchy.

Let P denote a convex polytope in \mathbb{R}^d . Each level i in the hierarchy corresponds to a δ_i -approximation of the boundary of P by a set of $O(1/\delta_i^{(d-1)/2})$ ellipsoids, where $\delta_i = \Theta(1/2^i)$. Each ellipsoid is sandwiched between two Macbeath regions and has $O(1)$ children, which correspond to the ellipsoids of the following level that approximate the same portion of the boundary (see Figure 1). The hierarchy starts with $\delta_0 = \Theta(1)$ and stops after $O(\log \frac{1}{\delta})$ levels when $\delta_i = \delta$, for a desired approximation δ . We present a simple algorithm to construct the hierarchy in $O(n + 1/\delta^{3(d-1)/2})$ time. The polytope P can be presented as either the intersection of n halfspaces or the convex hull of n points. We present the relevant background in Section 3.

Our algorithm to compute an ε -kernel in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2 + \alpha})$ (Theorem 1.1) is conceptually quite simple. Since the time to build the ε -approximation hierarchy for the convex hull is prohibitively high, we use an approximation parameter $\delta = \varepsilon^{1/3}$ to build a δ -approximation hierarchy in $O(n + 1/\varepsilon^{(d-1)/2})$ time. By navigating through this hierarchy, we partition the n points among the leaf Macbeath ellipsoids in $O(n \log \frac{1}{\varepsilon})$ time, discarding points that are too far from the boundary. We then compute an (ε/δ) -kernel for the set of points in each leaf ellipsoid and return the union of the kernels computed.

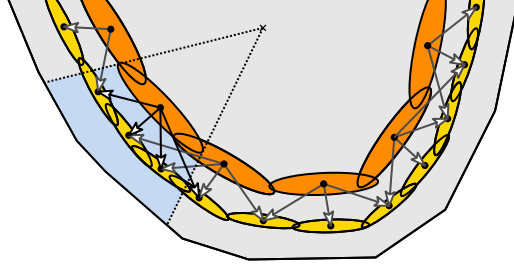


Figure 1: Two levels of the hierarchy of ellipsoids based on Macbeath regions.

Given an algorithm to compute an ε -kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t(d-1)})$ time, the previous procedure produces an ε -kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t'(d-1)})$ time where $t' = (4t + 1)/6$. Bootstrapping the construction a constant number of times, the value of t goes down from 1 to a value that is arbitrarily close to $1/2$. This discrepancy accounts for the $O(1/\varepsilon^\alpha)$ factors in our running times. In Section 4, we present the complete algorithm and its analysis, proving Theorem 1.1.

In Section 5, we use our kernel construction in the dual space to efficiently build a polytope membership data structure, proving Theorem 1.5. The key idea is to compute multiple kernels in order to avoid examining the whole polytope when building each Macbeath region. Again, we use bootstrapping to obtain a near-optimal preprocessing time. The remaining theorems follow from Theorems 1.1 and 1.5, together with several known reductions (Section 6).

2 Geometric Preliminaries

Consider a convex body K in d -dimensional space \mathbb{R}^d . Let ∂K denote the boundary of K . Let O denote the origin of \mathbb{R}^d . Given a parameter $0 < \gamma \leq 1$, we say that K is γ -fat if there exist concentric Euclidean balls B and B' , such that $B \subseteq K \subseteq B'$, and $\text{radius}(B)/\text{radius}(B') \geq \gamma$. We say that K is fat if it is γ -fat for a constant γ (possibly depending on d , but not on ε).

Unless otherwise specified, the notion of ε -approximation between convex bodies will be based on the direction-sensitive definition given in Section 1.2. We say that a convex body K' is an *absolute* ε -approximation to another convex body K if they are within Hausdorff error ε of each other. Further, we say that K' is an *inner* (resp., *outer*) approximation if $K' \subseteq K$ (resp., $K' \supseteq K$).

Let B_0 denote a ball of radius $r_0 = 1/2$ centered at the origin. For $0 < \gamma \leq 1$, let γB_0 denote the concentric ball of radius $\gamma r_0 = \gamma/2$. We say that a convex body K is in γ -canonical form if it is nested between γB_0 and B_0 . A body in γ -canonical form is γ -fat and has diameter $\Theta(1)$. We will refer to point O as the *center* of P .

For any point $x \in K$, define $\delta(x)$ to be minimum distance from x to any point on ∂K . For the sake of ray-shooting queries, it is useful to define a ray-based notion of distance as well. Given $x \in K$, define the *ray-distance* of x to the boundary, denoted $\text{ray}(x)$, as follows. Consider the intersection point p of ∂K and the ray emanating from O and passing through x . We define $\text{ray}(x) = \|xp\|$. The following utility lemma will be helpful in relating distances to the boundary.

Lemma 2.1. *Given a convex body K in γ -canonical form:*

- (a) *For any point $x \in P$, $\delta(x) \leq \text{ray}(x) \leq \delta(x)/\gamma$.*
- (b) *Let h be a supporting hyperplane of K . Let p be any point inside K at distance at most w from h , where $w \leq \gamma/4$. Let p' denote the intersection of the ray Op and h . Then $\|pp'\| \leq 2w/\gamma$.*
- (c) *Let p be any point on the boundary of K , and let h be a supporting hyperplane at p . Let h' denote the hyperplane obtained by translating h in the direction of the outward normal by w . Let p' denote the intersection of the ray Op with h' . Then $\|pp'\| \leq w/\gamma$.*

We omit the straightforward proof. The lower bound on $\text{ray}(x)$ for part (a) is trivial, and the upper bound follows by a straightforward adaption of Lemma 4.2 of [7]. Part (b) is an adaption of Lemma 2.11 of [8], and part (c) is similar.

For any centrally symmetric convex body A , define A^λ to be the body obtained by scaling A by a factor of λ about its center. The following lemma appears in Barany [13].

Lemma 2.2. *Let $\lambda \geq 1$. Let A and B be centrally symmetric convex bodies such that $A \subseteq B$. Then $A^\lambda \subseteq B^\lambda$.*

2.1 Caps and Macbeath Regions

Much of the material in this section has been presented in [7, 8]. We include it here for the sake of completeness. Given a convex body K , a *cap* C is defined to be the nonempty intersection of K with a halfspace (see Figure 2(a)). Let h denote the hyperplane bounding this halfspace. We define the *base* of C to be $h \cap K$. The *apex* of C is any point in the cap such that the supporting hyperplane of K at this point is parallel to h . The *width* of C , denoted $\text{width}(C)$, is the distance between h and this supporting hyperplane. Given any cap C of width w and a real $\lambda \geq 0$, we define its λ -*expansion*, denoted C^λ , to be the cap of K cut by a hyperplane parallel to and at distance λw from this supporting hyperplane. (Note that $C^\lambda = K$, if λw exceeds the width of K along the defining direction.)

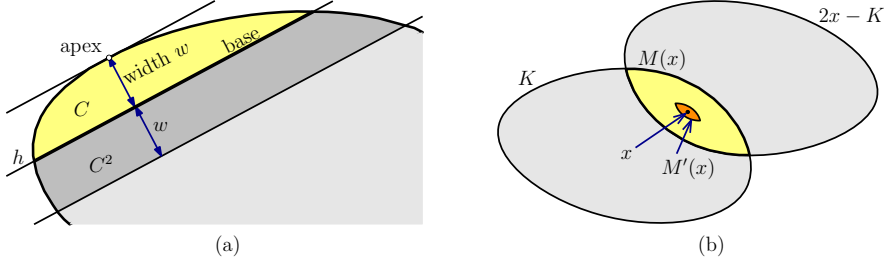


Figure 2: (a) Cap concepts and (b) Macbeath regions.

Given a point $x \in K$ and real parameter $\lambda \geq 0$, the *Macbeath region* $M^\lambda(x)$ (also called an *M-region*) is defined as:

$$M^\lambda(x) = x + \lambda((K - x) \cap (x - K)).$$

It is easy to see that $M^1(x)$ is the intersection of K and the reflection of K around x (see Figure 2(b)). Clearly, $M^1(x)$ is centrally symmetric about x , and $M^\lambda(x)$ is a scaled copy of $M^1(x)$ by the factor λ about x . We refer to x as the *center* of $M^\lambda(x)$ and to λ as its *scaling factor*. As a convenience, we define $M(x) = M^1(x)$ and $M'(x) = M^{1/5}(x)$. We refer to the latter as the *shrunk* Macbeath region.

We now present a few lemmas that encapsulate key properties of Macbeath regions. The first lemma shows that if two shrunk Macbeath regions have a nonempty intersection, then a constant factor expansion of one contains the other [8, 17, 24].

Lemma 2.3. *Let K be a convex body, and let $\lambda \leq 1/5$ be any real. If $x, y \in K$ such that $M^\lambda(x) \cap M^\lambda(y) \neq \emptyset$, then $M^\lambda(y) \subseteq M^{4\lambda}(x)$.*

The next lemma is useful in situations when we know that a shrunk Macbeath region partially overlaps a cap of K . It allows us to conclude that a constant factor expansion of the cap will fully contain the Macbeath region. The proof appears in [7].

Lemma 2.4. *Let K be a convex body. Let C be a cap of K and x be a point in K such that $C \cap M'(x) \neq \emptyset$. Then $M'(x) \subseteq C^2$.*

The following lemma shows that all points in a shrunk Macbeath region have similar distances from the boundary of K . The proof appears in [8].

Lemma 2.5. *Let K be a convex body. If $x \in K$ and $x' \in M'(x)$, then $4\delta(x)/5 \leq \delta(x') \leq 4\delta(x)/3$.*

For any $\delta > 0$, define the δ -erosion of a convex body K , denoted $K(\delta)$, to be the closed convex body formed by removing from K all points lying within distance δ of ∂K . The next lemma bounds the number of disjoint Macbeath regions that can be centered on the boundary of $K(\delta)$. The proof appears in [8].

Lemma 2.6. Consider a convex body $K \subset \mathbb{R}^d$ in γ -canonical form for some constant γ . Define $\Delta_0 = \frac{1}{2}(\gamma^2/(4d))^d$. For any fixed constant $0 < \lambda \leq 1/5$ and real parameter $\delta \leq \Delta_0$, let \mathcal{M} be a set of disjoint λ -scaled Macbeath regions whose centers lie on the boundary of $K(\delta)$. Then $|\mathcal{M}| = O(1/\delta^{(d-1)/2})$.

2.2 Shadows of Macbeath regions

Shrunken Macbeath regions reside within the interior of the convex body, but it is useful to identify the portion of the body's boundary that this Macbeath region will be responsible for approximating. For this purpose, we introduce the shadow of a Macbeath region. Given a convex body K that contains the origin O and a region $R \subseteq K$, we define the *shadow* of R (with respect to K), denoted $\text{shadow}(R)$, to be the set of points $x \in K$ such that the line segment Ox intersects R .

We also define a set of *normal directions* for R , denoted $\text{normals}(R)$. Consider the set of all hyperplanes that support K at some point in the shadow of R . Define $\text{normals}(R)$ to be the set of outward unit normals to these supporting hyperplanes. Typically, the region R in our constructions will be a (scaled) Macbeath region or an associated John ellipsoid (as defined in Section 3), close to the boundary of K . The following lemma captures a salient feature of these shadows, namely, that the shadow of a Macbeath region $M'(x)$ can be enclosed in an ellipsoid whose width in all normal directions is $O(\delta(x))$.

Lemma 2.7. Let $K \subset \mathbb{R}^d$ be a convex body in γ -canonical form for some constant γ . Let x be a point at distance δ from the boundary of K , where $\delta \leq \Delta_0$. Let $M = M'(x)$, $S = \text{shadow}(M)$, $N = \text{normals}(M)$, and $\widehat{M} = M^{4/\gamma}(x)$. Then:

$$(a) S \subseteq \widehat{M}.$$

$$(b) \text{width}_v(S) \leq c_1 \delta \text{ for all } v \in N. \text{ Here } c_1 \text{ is the constant } 8/(3\gamma).$$

$$(c) \text{width}_v(\widehat{M}) \leq c_2 \delta \text{ for all } v \in N. \text{ Here } c_2 \text{ is the constant } 160/(3\gamma^2).$$

Proof. We first prove (a). Consider any point $p \in \partial K \cap S$. Let y denote the first point of intersection of the ray Op with the Macbeath region M . To prove (a), it suffices to show that the segment yp is contained in \widehat{M} which, by convexity, is equivalent to showing that both points y and p are contained in \widehat{M} . Since $y \in M$, we have $y \in \widehat{M}$. To prove that $p \in \widehat{M}$, observe that a straightforward consequence of the definition of Macbeath regions is that $M(y)$ must contain a ball of radius $\delta(y)$ centered at y . Further, by Lemma 2.1(a), $\|yp\| = \text{ray}(y) \leq \delta(y)/\gamma$. It follows that $p \in M^{1/\gamma}(y)$. Also, since $y \in M'(x)$, it follows trivially that $M'(y)$ overlaps with $M'(x)$. Thus, by Lemma 2.3, $M'(y) \subseteq M^{4/5}(x)$. Applying Lemma 2.2 to $M'(y)$ and $M^{4/5}(x)$ with $\lambda = 5/\gamma$, we obtain $M^{1/\gamma}(y) \subseteq M^{4/\gamma}(x) = \widehat{M}$. Thus $p \in \widehat{M}$, which proves (a).

To prove (b), let p be any point of $\partial K \cap S$ and let y denote any point in the intersection of the ray Op with $M'(x)$. Let h denote any hyperplane supporting K at p . Let v denote the outward normal to h . We translate hyperplane h to pass through y and let C denote the cap of K cut by the resulting hyperplane. Clearly $\text{width}(C) \leq \|py\|$. By Lemma 2.1(a), $\|py\| = \text{ray}(y) \leq \delta(y)/\gamma$. Since $y \in M'(x)$, by Lemma 2.5, $\delta(y) \leq 4\delta(x)/3 = 4\delta/3$. Thus $\text{width}(C) \leq \delta(y)/\gamma \leq 4\delta/(3\gamma)$. Also, by Lemma 2.4, since $M'(x)$ overlaps C , $M'(x) \subseteq C^2$. It follows from convexity that $S \subseteq C^2$ and thus

$$\text{width}_v(S) \leq \text{width}(C^2) \leq 2 \text{width}(C) \leq 8\delta/(3\gamma),$$

which proves (b).

To prove (c), recall that $M'(x) \subseteq S$, which implies that $\text{width}_v(M'(x)) \leq 8\delta/(3\gamma)$. Thus $\text{width}_v(\widehat{M}) = (20/\gamma) \text{width}_v(M'(x)) \leq 160\delta/(3\gamma^2)$. \square

2.3 Representation Conversions

Convex sets are naturally described in two ways, as the convex hull of a discrete set of points and as the intersection of a discrete set of halfspaces. Some computational tasks are more easily performed using one operation or the other. For this reason, it will be useful to be able to convert between one representation and the other. Also, when approximate representations suffice, it will be useful

to prune a large set down to a smaller size. In this section we will present a few technical utilities to perform these conversions.

Given an n -element point set in \mathbb{R}^d , Chan showed that it is possible to construct an ε -kernel of size $O(1/\varepsilon^{(d-1)/2})$ in time $O(n + 1/\varepsilon^{d-1})$ [19]. The following lemma shows that, by applying Chan's construction, it is possible to efficiently approximate the convex hull of n points as the intersection of halfspaces.

Lemma 2.8. *Let $\gamma < 1$ be a positive constant, and $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ -canonical form represented as the convex hull of n points. In $O(n + 1/\varepsilon^{d-1})$ time it is possible to compute a polytope P' represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces such that P' is an inner absolute ε -approximation of P .*

Proof. Throughout the proof, to avoid tracking the numerous constant factors, we use the notation $O(\varepsilon)$ to denote a quantity that is a suitable scaling of ε by a constant factor. Let S be the input set of n points such that $P = \text{conv}(S)$. Since P is in γ -canonical form, we have $S \subseteq B_0$, where B_0 denotes the ball of radius $r_0 = 1/2$ centered at the origin. By applying Chan's kernel construction [19], in time $O(n + 1/\varepsilon^{d-1})$ we can compute a point set S'' of size $O(1/\varepsilon^{(d-1)/2})$ such that $\text{conv}(S'')$ is an absolute $O(\varepsilon)$ -approximation of $\text{conv}(S)$.

We then apply the polar transformation to the points of S'' yielding a set of $O(1/\varepsilon^{(d-1)/2})$ halfspaces in the dual space. It follows from standard properties of the polar transformation that the polytope \hat{P} defined by the intersection of these halfspaces is fat and has constant diameter (see, e.g., Lemma 7.2 of the journal version of [5]). This can be performed in time $O(1/\varepsilon^{(d-1)/2})$.

Next, take a sufficiently large hypercube centered at the origin that contains \hat{P} and there is constant separation between the boundary of this hypercube and \hat{P} . (Side length $O(1/\gamma)$ suffices.) We superimpose a grid of side length $\Theta(\sqrt{\varepsilon})$ on each of the $2d$ facets of this hypercube. Letting G denote the resulting set of grid points on the boundary of the hypercube, we have $|G| = O(1/\varepsilon^{(d-1)/2})$. Through the use of quadratic programming we compute the nearest neighbor of each point of G on the boundary of \hat{P} . For each grid point this can be done in time linear in the number of halfspaces that define \hat{P} [28, 29]. Thus, the total time for computing all the nearest neighbors is $O(|S''| \cdot |G|) = O(1/\varepsilon^{d-1})$. Letting \hat{S} denote the set of nearest neighbors so obtained, we have $|\hat{S}| = O(1/\varepsilon^{(d-1)/2})$. By standard results, $\text{conv}(\hat{S})$ is an absolute $O(\varepsilon)$ -approximation of \hat{P} [18, 22].

We again apply the polar transformation, mapping the set \hat{S} back to the primal space to obtain a set H of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. Let P'' be the polytope formed by intersecting these halfspaces. Recalling that $\text{conv}(\hat{S})$ is an absolute $O(\varepsilon)$ -approximation of \hat{P} , it follows from standard results that P'' is an absolute $O(\varepsilon)$ -approximation of $\text{conv}(S'')$. This step takes time $O(1/\varepsilon^{(d-1)/2})$.

Since P'' is an absolute $O(\varepsilon)$ -approximation of $\text{conv}(S'')$, and $\text{conv}(S'')$ is an absolute $O(\varepsilon)$ -approximation of P , it follows that (subject to a suitable choice of constant factors) P'' is an absolute $O(\varepsilon)$ -approximation of P . Define P' to be the polytope obtained by first translating the bounding halfspaces of P'' (i.e., the halfspaces of H) towards the origin by an amount $\Theta(\varepsilon)$ and then intersecting the resulting halfspaces. Clearly, P' is then an absolute inner $O(\varepsilon)$ -approximation of P , as desired.

The overall running time is dominated by the time needed to compute the kernel, and the time needed to compute the nearest neighbors for the points of G . \square

The following lemma is useful when representing polytopes by the intersection of halfspaces.

Lemma 2.9. *Let $\gamma < 1$ be a positive constant, and $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ -canonical form represented as the intersection of n halfspaces. In $O(n + 1/\varepsilon^{d-1})$ time it is possible to compute a polytope P' represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces such that P' is an outer absolute ε -approximation of P .*

Proof. Let H denote the set of n halfspaces defining P . We apply the polar transformation to the halfspaces of H obtaining a set S of n points in the dual space. It follows from standard properties of the polar transformation that $\text{conv}(S)$ is fat and has constant diameter (see, e.g., the journal version of [5]). This step can be performed in $O(n)$ time. By applying Chan's kernel construction [19], in time $O(n + 1/\varepsilon^{d-1})$, we can compute a point set S' of size $O(1/\varepsilon^{(d-1)/2})$ such that $\text{conv}(S')$ is an inner absolute $O(\varepsilon)$ -approximation of $\text{conv}(S)$. We again apply the polar transformation, mapping the set S' back to the primal space to obtain a set H' of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. Let P' be the polytope

formed by intersecting these halfspaces. Since $\text{conv}(S')$ is an inner absolute $O(\varepsilon)$ -approximation of $\text{conv}(S)$, it follows that (subject to a suitable choice of constant factors), P' is an absolute outer $O(\varepsilon)$ -approximation of P , as desired. The total time is dominated by the time to compute the kernel. \square

Remark: Theorem 1.1 shows that an ε -kernel of size $O(1/\varepsilon^{(d-1)/2})$ can be computed in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$. The construction time in Lemma 2.9 is asymptotically the same as the time needed to construct an ε -kernel. Therefore, the construction time can be reduced to this quantity.

3 Hierarchy of Macbeath Ellipsoids

The data structure presented in [8] for the approximate polytope membership problem is based on constructing a hierarchy of ellipsoids. In this section, we present a variant of this structure, which will play an important role in our constructions.

For a Macbeath region $M^\lambda(x)$, we denote its circumscribing John ellipsoid by $E^\lambda(x)$, which we call a *Macbeath ellipsoid*. Since Macbeath regions are centrally symmetric and the constant in John's Theorem [25] is \sqrt{d} for centrally symmetric bodies, we have $E^\lambda(x) \subseteq M^{\lambda\sqrt{d}}(x)$. Recall the constant $\Delta_0 = \frac{1}{2}(\gamma^2/4d)^d$ defined in the statement of Lemma 2.6, and define $\lambda_0 = 1/(20d)$. Each level of our structure is based on the following lemma. (We caution the reader that in the lemmas of this section, the value of n used in the application of the lemma may differ from the original input size.)

Lemma 3.1. *Let $\gamma < 1$ be a positive constant, and let $0 < \delta \leq \Delta_0$ be a real parameter. Let P be a polytope in γ -canonical form, represented as the intersection of n halfspaces. There exists a set $X \subseteq \partial P(\delta)$ consisting of $O(1/\delta^{(d-1)/2})$ points such that the following properties hold:*

- (a) *The set of Macbeath regions $\{M^{\lambda_0}(x) : x \in X\}$ are pairwise disjoint.*
- (b) *The set of Macbeath ellipsoids $\{E^{4\lambda_0\sqrt{d}}(x) : x \in X\}$ together cover $\partial P(\delta)$.*

Furthermore, in $O(n/\delta^{d-1} + 1/\delta^{3(d-1)/2})$ time, we can construct the set of Macbeath ellipsoids $\{E^{4\lambda_0\sqrt{d}}(x) : x \in X\}$.

Proof. We first show how to construct the required set of Macbeath ellipsoids. Translate each bounding halfspace of P towards the origin by amount δ . It is easy to see that the polytope $P(\delta)$ is the intersection of the translated halfspaces. This can be done in $O(n)$ time.

Recalling that $P \subseteq B_0$, where B_0 is the ball of radius $r_0 = 1/2$, consider the hypercube just enclosing B_0 . Superimpose a $\Theta(\delta)$ -grid on each of the $2d$ facets of this hypercube. Intersect the segment joining the origin to each grid point with $\partial P(\delta)$, and let $X' \subset \partial P(\delta)$ denote the resulting set of intersection points. Note that $|X'| = O(1/\delta^{d-1})$. Using the fact that $P(\delta)$ is fat, a straightforward geometric calculation shows that for any point on $\partial P(\delta)$, there is a point of X' within distance $c\delta$ of it, where c is a suitable constant. (Adjusting the constant factor in the grid spacing, we can ensure that $c \leq \lambda_0\sqrt{d}$, which is a fact that we will use later in the proof.) As each point of X' can be determined in $O(n)$ time, X' can be computed in $O(n/\delta^{d-1})$ time.

For each $x' \in X'$, construct $M^{\lambda_0}(x')$. Let \mathcal{M}' denote the resulting set of Macbeath regions. As a straightforward consequence of the definition of Macbeath regions, we can compute each Macbeath region in $O(n)$ time (i.e., in time proportional to the number of halfspaces that define P). Note that we represent each Macbeath region as the intersection of n halfspaces. For each Macbeath region $M^{\lambda_0}(x') \in \mathcal{M}'$, determine the circumscribing John ellipsoid $E^{\lambda_0}(x')$. By standard results, we can construct the John ellipsoid of a convex polytope in time that is linear in the number of its defining halfspaces [21]. Thus, this step also takes time $O(n|\mathcal{M}'|) = O(n/\delta^{d-1})$.

Next, we will determine a maximal subset $\mathcal{M} \subseteq \mathcal{M}'$ such that the John ellipsoids associated with the Macbeath regions of \mathcal{M} are disjoint. Initialize $\mathcal{M} = \emptyset$. Examine the Macbeath regions of \mathcal{M}' one by one. Insert the Macbeath region into \mathcal{M} if its associated John ellipsoid does not intersect the John ellipsoid of any Macbeath region of \mathcal{M} . Clearly, this method yields a maximal subset $\mathcal{M} \subseteq \mathcal{M}'$ such that the associated John ellipsoids are disjoint. To bound the time required for this step, observe that the Macbeath regions of \mathcal{M} are disjoint, and so by Lemma 2.6, $|\mathcal{M}| = O(1/\delta^{(d-1)/2})$. Since it takes constant time to check whether two ellipsoids intersect, it follows that the time required is $O(|\mathcal{M}'| \cdot |\mathcal{M}|) = O(1/\delta^{3(d-1)/2})$.

Finally, to obtain the desired ellipsoids let X denote the set of centers of the Macbeath regions of \mathcal{M} . For each $x \in X$, we scale the associated ellipsoid $E^{\lambda_0}(x)$ constructed above about its center by a factor of $4\sqrt{d}$ to obtain the ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$. This step can be done in time $O(|\mathcal{M}|) = O(1/\delta^{(d-1)/2})$.

By combining the time of the above steps we obtain the desired overall construction time of $O(n/\delta^{d-1} + 1/\delta^{3(d-1)/2})$. The bound on $|X|$ follows from the bound on \mathcal{M} . By our earlier remarks, the set of Macbeath regions $\{M^{\lambda_0}(x) : x \in X\}$ are pairwise disjoint, which proves Property (a).

It remains to establish Property (b), that is, to show that the union of the ellipsoids $\{E^{4\lambda_0\sqrt{d}}(x) : x \in X\}$ covers $\partial P(\delta)$. Towards these end, consider any point $p \in \partial P(\delta)$. We will show that there is an $x \in X$ such that the ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$ contains p . Recall that there is a point $x' \in X'$ that is within distance $c\delta$ of p , where c is a constant no more than $\lambda_0\sqrt{d}$. A straightforward consequence of the definition of Macbeath regions is that, for any point $y \in P$, the Macbeath region $M(y)$ contains a ball of radius $\delta(y)$ centered at y . It follows that $M^{\lambda_0\sqrt{d}}(x')$ contains a ball of radius $\lambda_0\sqrt{d}\delta$ centered at x' . Hence, $p \in M^{\lambda_0\sqrt{d}}(x') \subseteq M^{4\lambda_0\sqrt{d}}(x') \subseteq E^{4\lambda_0\sqrt{d}}(x')$. Thus, if $x' \in X$, we are done.

We next consider the case when $x' \notin X$. In this case, it follows from our construction that there is an $x \in X$ such that $E^{\lambda_0}(x)$ intersects $E^{\lambda_0}(x')$. Recall that $E^{\lambda_0}(x) \subseteq M^{\lambda_0\sqrt{d}}(x)$ and $E^{\lambda_0}(x') \subseteq M^{\lambda_0\sqrt{d}}(x')$. Thus $M^{\lambda_0\sqrt{d}}(x)$ intersects $M^{\lambda_0\sqrt{d}}(x')$. Applying Lemma 2.3, it follows that $M^{\lambda_0\sqrt{d}}(x') \subseteq M^{4\lambda_0\sqrt{d}}(x)$. Thus $p \in M^{4\lambda_0\sqrt{d}}(x) \subseteq E^{4\lambda_0\sqrt{d}}(x)$. It follows that the ellipsoids $E^{4\lambda_0\sqrt{d}}(x)$, for $x \in X$, together cover $\partial P(\delta)$, which completes the proof of the lemma. \square

Based on the above lemma, we are now ready to describe our hierarchical data structure. Let P be a polytope in γ -canonical form, where γ is a constant. Recall the constant $\Delta_0 = \frac{1}{2}(\gamma^2/(4d))^d$, and for $i \geq 0$ define $\Delta_i = \Delta_0/2^i$. The data structure consists of levels $0, 1, \dots, \ell$, where ℓ is the smallest integer such that $\Delta_\ell \leq \delta$. Since γ is a constant, $\ell = O(\log \frac{1}{\delta})$. Since P is in γ -canonical form, the origin O is at distance at least $\gamma/2$ from ∂P . Since $\Delta_0 < \gamma/2$, it follows that $P(\Delta_0)$ contains O and $P(\Delta_i) \subset P(\Delta_{i+1})$ for $0 \leq i \leq \ell - 1$. For each level i , we apply Lemma 3.1, setting δ in the lemma to Δ_i . In $O(n/\Delta_i^{d-1} + 1/\Delta_i^{3(d-1)/2})$ time, we obtain a set of $O(1/\Delta_i^{(d-1)/2})$ ellipsoids that cover $\partial P(\Delta_i)$. Summing over all levels, the number of ellipsoids is $O(1/\delta^{(d-1)/2})$ and the time taken is $O(n/\delta^{d-1} + 1/\delta^{3(d-1)/2})$.

Our data structure is a directed acyclic graph (DAG), where the nodes at level i correspond to the ellipsoids computed for level i . The children of an ellipsoid E at level i are the ellipsoids E' at level $i+1$ such that there exists a ray from the origin that simultaneously intersects E and E' . Since this is a constant time operation for any two given ellipsoids, it takes $O(1/\delta^{d-1})$ time to find the children of all the nodes in the DAG. It is convenient to root the DAG by creating a special node whose children are all the nodes of level zero. An important property of the hierarchy is that each node only has $O(1)$ children. The proof of this property is similar to that given in [8]. We include the proof for the sake of completeness. For the root, this follows from the fact that the number of nodes of level zero is $O(1/\Delta_0^{(d-1)/2})$ and Δ_0 is a constant. For non-root nodes, the proof is based on the following lemma, which is proved in [8].

Lemma 3.2. *Let $K \subset \mathbb{R}^d$ be a convex body in γ -canonical form for some constant γ , and let $\lambda \leq 1/5$ be any constant. Let $x \in K$ such that $\delta(x) \leq \Delta_0$. Consider the generalized cone formed by rays emanating from the center O of K and intersecting $M'(x)$. Let Y denote any set of points y such that $\delta(y) = \delta(x)/2$ and the set of Macbeath regions $M^\lambda(y)$ are disjoint. Let $Y' \subseteq Y$ denote the set of points y such that $M'(y)$ overlaps the aforementioned cone. Then $|Y'| = O(1)$.*

For any node w , we let x_w denote the center of the associated ellipsoid. Consider any node u at level $i \geq 0$. Also, consider the generalized cone formed by rays emanating from the origin that intersect the ellipsoid $E^{4\lambda_0\sqrt{d}}(x_u)$ associated with u . The children of u are those nodes v at level $i+1$ whose ellipsoid $E^{4\lambda_0\sqrt{d}}(x_v)$ intersects this generalized cone.

Since $x_u \in \partial P(\Delta_i)$, we have $\delta(x_u) = \Delta_i \leq \Delta_0$. Recall that $E^{4\lambda_0\sqrt{d}}(x_u) \subseteq M^{4\lambda_0 d}(x_u) = M'(x_u)$. Thus, the generalized cone of rays that intersect $M'(x_u)$ includes all the rays used to define the children of u . The points x_v that form level $i+1$ of the structure lie on $\partial P(\Delta_{i+1})$ and thus satisfy $\delta(x_v) = \delta(x_u)/2$. By Property (a) of Lemma 3.1, the Macbeath regions $M^{\lambda_0}(x_v)$ are pairwise disjoint, thus they constitute a set Y as described in the preconditions of Lemma 3.2. Each child

v of u corresponds to a point x_v such that the ellipsoid $E^{4\lambda_0\sqrt{d}}(x_v)$ intersects the generalized cone. Reasoning as we did above for x_u , we have $E^{4\lambda_0\sqrt{d}}(x_v) \subseteq M'(x_v)$. Therefore, the points x_v associated with the children of u constitute a subset of the set Y' given in the lemma. Therefore, the number of children of u is $O(1)$, as desired.

Given a query ray Oq , our data structure allows us to quickly find a leaf node such that the associated ellipsoid intersects this ray. The query algorithm descends the DAG by starting at the root and visiting any node at level zero that intersects the ray. Letting u denote the current node, we next visit any child of u whose associated ellipsoid intersects the ray. We repeat this procedure until a leaf node is reached. As the number of levels is $O(\log \frac{1}{\delta})$, this quantity bounds the time taken by this procedure. We summarize the main result of this section.

Lemma 3.3. *Let $\gamma < 1$ be a positive constant, and let $0 < \delta \leq \Delta_0$ be a real parameter. Let P be a polytope in γ -canonical form, represented as the intersection of n halfspaces. In $O(n/\delta^{d-1} + 1/\delta^{3(d-1)/2})$ time, we can construct the DAG structure described above. In particular, the DAG satisfies the following properties:*

- (a) *The total number of nodes (including leaves), and the total space used by the DAG are both $O(1/\delta^{(d-1)/2})$.*
- (b) *Each leaf is associated with an ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$, where $x \in \partial P(\delta)$. The union of the ellipsoids associated with all the leaves covers $\partial P(\delta)$.*
- (c) *Given a query ray Oq , in $O(\log \frac{1}{\delta})$ time, we can find a leaf node such that the associated ellipsoid intersects this ray.*

Given a convex body K and query point q , an *absolute ε -APM* query returns a positive result if q lies within K , a negative result if q is at distance at least ε from K , and otherwise it may return either result. After a small enhancement, this DAG can be used for answering absolute ε -APM queries for a polytope P in γ -canonical form. We assume that P is represented as the intersection of a set H of n halfspaces. We invoke the above lemma for $\delta = \varepsilon\gamma/(2c_1)$, where c_1 is the constant of Lemma 2.7(b). We then associate each leaf of the DAG with a halfspace as follows. Let x denote the center of the leaf ellipsoid and let p denote the intersection of the ray Ox with ∂P . Let $h \in H$ denote any supporting halfspace of P (containing P) at p . We store h with this leaf. By exhaustive search, we can determine h in $O(n)$ time, so the total time for this step is $O(n/\varepsilon^{(d-1)/2})$. Asymptotically, this does not affect the time it takes to construct the data structure. Given a query point q , we answer queries by first determining a leaf whose ellipsoid intersects the ray Oq . By Lemma 3.3(c), this takes $O(\log \frac{1}{\varepsilon})$ time. We return a positive answer if and only if q is contained in the associated halfspace. We establish the correctness of this method in the following lemma.

Lemma 3.4. *Given a query point q , the query procedure returns a valid answer to the absolute ε -APM query.*

Proof. Consider the leaf whose ellipsoid intersects the ray Oq . Let $E^{4\lambda_0\sqrt{d}}(x)$ denote the associated ellipsoid and let h be the halfspace stored with this leaf. Recall that h is a supporting halfspace at the point p where the ray Ox intersects ∂P . If $q \in P$ then clearly $q \in h$ and such a query point is correctly declared as lying inside P . To complete the proof, we need to show that if $q \notin P$ and the distance of q from ∂P is greater than ε , then $q \notin h$. In this case, q would be correctly declared as lying outside P .

Let y denote any point in the intersection of the ray Oq with the leaf ellipsoid. Let y' denote the intersection of the ray Oq with the hyperplane bounding h . To prove the claim, it suffices to show that $\|yy'\| \leq \varepsilon$. Recall that $E^{4\lambda_0\sqrt{d}}(x) \subseteq M^{4\lambda_0 d}(x) = M'(x)$. Let $M = M'(x)$, $S = \text{shadow}(M)$, and $N = \text{normals}(M)$. Clearly $y, p \in S$ and the normal vector v to the hyperplane bounding h belongs to N . By Lemma 2.7, $\text{width}_v(S) \leq c_1\delta(x) = \varepsilon\gamma/2$. Note that the distance of y from the hyperplane bounding h is at most $\text{width}_v(S)$. Applying Lemma 2.1(b), we obtain $\|yy'\| \leq (2/\gamma)(\varepsilon\gamma/2) = \varepsilon$, as desired. \square

We summarize the result below.

Lemma 3.5. *Let $\gamma < 1$ be a positive constant, and let $\varepsilon > 0$ be a real parameter. Let P be a polytope in γ -canonical form, represented as the intersection of n halfspaces. In $O(n/\varepsilon^{d-1} + 1/\varepsilon^{3(d-1)/2})$ time, we can construct a data structure that uses $O(1/\varepsilon^{(d-1)/2})$ space and answers absolute ε -APM queries in $O(\log \frac{1}{\varepsilon})$ time.*

4 Kernel Construction

In this section we show how to build an ε -kernel efficiently, proving Theorem 1.1. The input to an ε -kernel construction consists of the approximation parameter ε and a set S of n points. Our algorithm is based on a bootstrapping strategy. We assume that we have access to an algorithm that can construct an ε -kernel of $O(1/\varepsilon^{(d-1)/2})$ size in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$, where $\beta > 0$ is a parameter. Recall that the size of the kernel is asymptotically optimal in the worst case. We will present a method for improving the running time of this algorithm. Recall that Chan [19] gave an algorithm for constructing kernels of optimal size which runs in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$. We will use this algorithm to initialize our bootstrapping scheme with $\beta = 1/2$.

Our method is based on executing the following steps. It uses a parameter $\delta = \varepsilon^{1/3}$.

1. We begin by “fattening” the input point set S . Formally, we compute an affine transformation that maps S to S' , such that $\text{conv}(S')$ is in γ -canonical form for some constant γ . By standard results (see, e.g., the journal version of [5]), this affine transformation and the set S' can be computed in $O(n)$ time.
2. Use Lemma 2.8 to build a polytope P , represented as the intersection of $O(1/\delta^{(d-1)/2})$ halfspaces, such that P is an inner absolute δ -approximation of $\text{conv}(S')$. This step takes $O(n + 1/\delta^{d-1}) = O(n + 1/\varepsilon^{(d-1)/3})$ time.
3. Construct the DAG structure of Lemma 3.3 for polytope P using the parameter δ . Replacing n in the statement of the lemma by $O(1/\delta^{(d-1)/2})$, it follows that this step takes $O(1/\delta^{3(d-1)/2}) = O(1/\varepsilon^{(d-1)/2})$ time.
4. For each point $p \in S'$, in $O(\log \frac{1}{\delta})$ time, we find a leaf of the DAG such that the associated ellipsoid $E^{4\lambda_0\sqrt{d}}(x)$ intersects the ray Op . Recall that $x \in \partial P(\delta)$. In $O(1)$ additional time, we can determine whether p lies in the shadow of this ellipsoid (with respect to $\text{conv}(S')$). If so, we associate p with this ellipsoid, otherwise we discard it. By Lemma 3.3(c), it takes $O(\log \frac{1}{\delta})$ time to process each point, thus the time taken for processing all the points of S' is $O(n \log \frac{1}{\delta}) = O(n \log \frac{1}{\varepsilon})$.
5. For each leaf ellipsoid of the DAG, we build a $(c_3\varepsilon/\delta)$ -kernel for the points of S' that lie in its shadow, where c_3 is a suitably small constant that will be selected later. This kernel computation is done using the aforementioned algorithm that computes ε -kernels of point sets of size n in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$. The size of the $O(\varepsilon/\delta)$ -kernel computed for each shadow is $O((\delta/\varepsilon)^{(d-1)/2})$ and the time required is $O(n_i \log \frac{\delta}{\varepsilon} + (\delta/\varepsilon)^{(1/2+\beta)(d-1)})$, where n_i denotes the number of points of S' in the shadow. Summed over all the shadows, it follows that the total time required is

$$O\left(n \log \frac{\delta}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{(\frac{1}{2}+\beta)(d-1)}\right) = O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{(\frac{1}{2}+\frac{2\beta}{3})(d-1)}\right).$$

Here we have used the facts that each point of S' is assigned to at most one shadow and the total number of shadows, which is bounded by the number of leaves in the DAG, is $O(1/\delta^{(d-1)/2})$.

6. Let $S'' \subseteq S'$ be the union of the kernels computed in the previous step. Since the number of shadows is $O(1/\delta^{(d-1)/2})$ and the size of the kernel for each shadow is $O((\delta/\varepsilon)^{(d-1)/2})$, it follows that $|S''| = O(1/\varepsilon^{(d-1)/2})$. We apply the inverse of the affine transformation computed in Step 1 to the points of S'' , and output the resulting set of points as the desired ε -kernel for S .

We have shown that the size of the output kernel is $O(1/\varepsilon^{(d-1)/2})$, as desired. The running time of Step 5 dominates the time complexity. The next lemma establishes the correctness of this construction.

Lemma 4.1. *The construction yields an ε -kernel.*

Proof. Throughout this proof, for a given convex body K , we use $M_K(x)$, $E_K(x)$, and $\delta_K(x)$ to denote the quantities $M(x)$, $E(x)$, and $\delta(x)$ with respect to K . Let $P' = \text{conv}(S')$. By standard results on fattening, it suffices to show that $\text{conv}(S'')$ is an absolute $O(\varepsilon)$ -approximation of P' . Let v be an arbitrary direction. Consider the extreme point p of S' in direction v . Clearly $p \in \partial P'$. Recall that P is an inner δ -approximation of P' , and the ellipsoids associated with the leaves of the DAG cover the boundary of $P(\delta)$. Thus, there must be an ellipsoid $E = E_P^{4\lambda_0\sqrt{d}}(x)$, $x \in \partial P(\delta)$, such that p is assigned to the shadow of E in Step 4. Note that this shadow and all shadows throughout this proof are assumed to be with respect to the polytope P' (and not P). We claim that $\text{width}_v(\text{shadow}(E)) \leq 2c_1\delta$, where c_1 is the constant of Lemma 2.7(b). Assuming this claim for now, let us complete the proof of the lemma. Recall that in Step 5, we built a $(c_3\varepsilon/\delta)$ -kernel for all the points of S' that are assigned to the shadow of E , and S'' includes all the points of this kernel. It follows that the distance between the supporting hyperplanes of $\text{conv}(S')$ and $\text{conv}(S'')$ in direction v is at most $(c_3\varepsilon/\delta) \cdot \text{width}_v(\text{shadow}(E)) \leq (c_3\varepsilon/\delta) \cdot (2c_1\delta) = 2c_1c_3\varepsilon$. By choosing c_3 sufficiently small, we can ensure that this quantity is smaller than any desired constant times ε , which proves the lemma.

It remains to show that $\text{width}_v(\text{shadow}(E)) \leq 2c_1\delta$. Recall that

$$E = E_P^{4\lambda_0\sqrt{d}}(x) \subseteq M_P^{4\lambda_0d}(x) = M_{P'}(x).$$

Furthermore, since $P \subseteq P'$, a straightforward consequence of the definition of Macbeath regions is that $M_P(x) \subseteq M_{P'}(x)$. To simplify the notation, let M denote $M_{P'}(x)$. Putting it together, we obtain $E \subseteq M$. Thus $\text{shadow}(E) \subseteq \text{shadow}(M)$, which implies that $\text{width}_v(\text{shadow}(E)) \leq \text{width}_v(\text{shadow}(M))$. By Lemma 2.7(b),

$$\text{width}_v(\text{shadow}(M)) \leq c_1\delta_{P'}(x).$$

Using the triangle inequality and the fact that P is an inner δ -approximation of P' , we obtain $\delta_{P'}(x) \leq \delta_P(x) + \delta = 2\delta$. Thus $\text{width}_v(\text{shadow}(E)) \leq \text{width}_v(\text{shadow}(M)) \leq 2c_1\delta$, as desired. \square

We are now ready to prove Theorem 1.1.

Proof. Our proof is based on a constant number of applications of the algorithm from this section. It suffices to show that there is an algorithm that can construct an ε -kernel of $O(1/\varepsilon^{(d-1)/2})$ size in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta')(d-1)})$, where $\beta' = \alpha/(d-1)$.

We initialize the bootstrapping process by Chan's algorithm [19], which has $\beta = 1/2$. Observe that the value of β is initially $1/2$ and falls by a factor of $2/3$ with each application of the algorithm. It follows that after $O(\log \frac{1}{\alpha})$ applications, we will obtain an algorithm with the desired running time. This completes the proof. \square

5 Approximate Polytope Membership

In this section we show how to obtain a data structure for approximate polytope membership, proving Theorem 1.5. Our best data structure for APM achieves query time $O(\log \frac{1}{\varepsilon})$ with storage $O(1/\varepsilon^{(d-1)/2})$ and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$. As with kernels, our construction here is again based on a bootstrapping strategy. To initialize the process, we will use a data structure that achieves the aforementioned query time with the same storage but with preprocessing time $O(n + 1/\varepsilon^{3(d-1)/2})$. The data structure is based on Lemma 3.5. Recall that the input is a polytope represented as the intersection of n halfspaces.

We begin by “fattening” the input polytope. Formally, we use an affine transformation to map the input polytope to a polytope P' that is in γ -canonical form. This step takes $O(n)$ time [5]. By standard results, it suffices to build a data structure for answering absolute $O(\varepsilon)$ -APM queries with respect to P' (see, e.g., Lemma 7.1 of the journal version of [5]).

Next, we apply Lemma 2.9 to construct an outer absolute $O(\varepsilon)$ -approximation P of P' , where P is represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. This step takes $O(n + 1/\varepsilon^{d-1})$ time. Finally, we use Lemma 3.5 to construct a data structure for answering absolute $O(\varepsilon)$ -APM queries

with respect to P . Replacing n in the statement of the lemma by $O(1/\varepsilon^{(d-1)/2})$, it follows that this step takes $O(1/\varepsilon^{3(d-1)/2})$ time.

The total construction time is $O(n + 1/\varepsilon^{3(d-1)/2})$. To answer a query, we map the query point using the same transformation used to fatten the polytope, and then use the data structure constructed above to determine whether the resulting point lies in polytope P . Subject to an appropriate choice of constant factors, the correctness of this method follows from the fact that P is an outer absolute $O(\varepsilon)$ -approximation of P' .

We summarize this result in the following lemma.

Lemma 5.1. *Let $\varepsilon > 0$ be a real parameter and let P be a polytope, represented as the intersection of n halfspaces. In $O(n + 1/\varepsilon^{3(d-1)/2})$ time, we can construct a data structure that uses $O(1/\varepsilon^{(d-1)/2})$ space and answers ε -APM queries in $O(\log \frac{1}{\varepsilon})$ time.*

We can now present the details of our bootstrapping approach. We assume that we have access to a data structure that can answer ε -APM queries in $O(\log \frac{1}{\varepsilon})$ time with $O(1/\varepsilon^{(d-1)/2})$ storage and $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta)(d-1)})$, where $\beta > 0$ is a parameter. We present a method for constructing a new data structure which matches the given data structure in space and query time, but has a lower preprocessing time. Our method uses a parameter $\delta = \varepsilon^{\beta/(1+\beta)}$.

1. As in the construction given above, we first fatten the input polytope. Formally, we use an affine transformation to map the input polytope to a polytope P' that is in γ -canonical form. This step takes $O(n)$ time. By standard results, it suffices to build a data structure for answering absolute $O(\varepsilon)$ -APM queries with respect to P' .
2. Use Lemma 2.9 to construct an outer absolute $O(\varepsilon)$ -approximation P of P' , where P is represented as the intersection of $O(1/\varepsilon^{(d-1)/2})$ halfspaces. By the remark following Lemma 2.9, this step takes $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time.
3. Construct the DAG of Lemma 3.3 for polytope P using the parameter δ . Replacing n in the statement of the lemma by $O(1/\varepsilon^{(d-1)/2})$, it follows that this step takes $O((1/\delta)^{d-1} \cdot (1/\varepsilon)^{(d-1)/2})$ time.
4. For each leaf of the DAG, we construct an APM data structure as follows. Let $E = E^{4\lambda_0\sqrt{d}}(x)$ denote the ellipsoid associated with the leaf. Let R denote the minimum enclosing hyperrectangle of the ellipsoid $E^{4/\gamma}(x)$. We will see later that R contains the shadow of E (with respect to P), and its width in any direction in $\text{normals}(E)$ is at most $c_2 d \delta = O(\delta)$, where c_2 is the constant in Lemma 2.7(c). We use the aforementioned algorithm for constructing an APM data structure for this region with approximation parameter $c_3 \varepsilon / \delta$, where c_3 is a sufficiently small constant that we will select later. Note that each such region can be expressed as the intersection of $n_i = O(1/\varepsilon^{(d-1)/2})$ halfspaces, namely, all the halfspaces defining P together with the $2d$ halfspaces defined by the facets of R . The construction time of the APM data structure for each leaf is

$$O\left(n_i \log \frac{\delta}{\varepsilon} + \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}} \log \frac{\delta}{\varepsilon} + \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right),$$

and the space used is $O((\delta/\varepsilon)^{(d-1)/2})$. Since there are $O(1/\delta^{(d-1)/2})$ leaves, it follows that the total space is $O(1/\varepsilon^{(d-1)/2})$, and the total construction time is the product of $O(1/\delta^{(d-1)/2})$ and the above construction time for each leaf.

Summing up the time over all the four steps, we get a total construction time on the order of

$$n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}+\alpha} + \left(\frac{1}{\delta}\right)^{d-1} \left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \cdot \left(\left(\frac{1}{\varepsilon}\right)^{\frac{d-1}{2}} \log \frac{\delta}{\varepsilon} + \left(\frac{\delta}{\varepsilon}\right)^{\left(\frac{1}{2}+\beta\right)(d-1)}\right).$$

Recalling that $\delta = \varepsilon^{\beta/(1+\beta)}$ and assuming that the constant α is much smaller than β , it follows that the construction time is

$$O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\left(\frac{1}{2}+\frac{\beta}{1+\beta}\right)(d-1)}\right).$$

We answer queries as follows. Recall the affine transformation used to fatten the input polytope. We apply this transformation on the input query point to obtain a point q . Recall that it suffices to answer absolute $O(\varepsilon)$ -APM queries for q with respect to P' . As P is an outer absolute $O(\varepsilon)$ -approximation of P' , it suffices to answer absolute $O(\varepsilon)$ -APM queries for q with respect to P . To answer this query, we identify a leaf of the DAG such that the associated ellipsoid E intersects the ray Oq . This takes time $O(\log \frac{1}{\delta})$. Let y denote an intersection point of this ray with the ellipsoid E . If q lies on the segment Oy , then q is declared as lying inside P . Otherwise we return the answer we get for query q using the APM data structure we built for this leaf. It takes time $O(\log \frac{\delta}{\varepsilon})$ to answer this query. Including the time to locate the leaf, the total query time is $O(\log \frac{1}{\varepsilon})$.

In Lemma 5.2, we show that queries are answered correctly.

Lemma 5.2. *The query procedure returns a valid answer to the ε -APM query.*

Proof. We borrow the terminology from the query procedure given above. As mentioned, it suffices to show that our algorithm correctly answers absolute $O(\varepsilon)$ -APM queries for q with respect to the polytope P . Recall that we identify a leaf of the DAG whose associated ellipsoid $E = E^{4\lambda_0\sqrt{d}}(x)$ intersects the ray Oq . Recall that y is a point on the intersection of the ray Oq with E . Clearly, if q lies on segment Oy , then $q \in P$ and q is correctly declared as lying inside P .

It remains to show that queries are answered correctly when $\|Oq\| > \|Oy\|$. In this case, we handle the query using the APM data structure we built for the leaf. Recall that this structure is built for the polytope formed by intersecting P with the smallest enclosing hyperrectangle R of the ellipsoid $E^{4/\gamma}(x)$. We claim that (i) $\text{shadow}(E) \subseteq R$ and (ii) $\text{width}_v(R) \leq c_2 d \delta$ for all $v \in \text{normals}(E)$, where c_2 is the constant in Lemma 2.7(c).

To see this claim, recall that $M^\lambda(x) \subseteq E^\lambda(x) \subseteq M^{\lambda\sqrt{d}}(x)$ for any $\lambda > 0$. Using this fact, it follows that $M^{4/\gamma}(x) \subseteq E^{4/\gamma}(x) \subseteq M^{4\sqrt{d}/\gamma}(x)$. By Lemma 2.7(a), $\text{shadow}(E) \subseteq M^{4/\gamma}(x)$. Thus $\text{shadow}(E) \subseteq E^{4/\gamma}(x) \subseteq R$, which proves (i). To prove (ii), note that $R \subseteq E^{4\sqrt{d}/\gamma}(x)$, since R is the smallest enclosing hyperrectangle of $E^{4/\gamma}(x)$. Also $E^{4\sqrt{d}/\gamma}(x) \subseteq M^{4d/\gamma}(x)$. Thus $R \subseteq M^{4d/\gamma}(x)$. By Lemma 2.7(c), $\text{width}_v(M^{4/\gamma}(x)) \leq c_2 \delta$ for all $v \in \text{normals}(M'(x))$. Since $R \subseteq M^{4d/\gamma}(x)$ and $E \subseteq M'(x)$, it follows that $\text{width}_v(R) \leq c_2 d \delta$ for all $v \in \text{normals}(E)$.

We return to showing that queries are correctly answered when $\|Oq\| > \|Oy\|$. We consider two possibilities depending on whether q is inside or outside P . If $q \in P$ then $q \in \text{shadow}(E)$. By part (i) of the above claim, $\text{shadow}(E) \subseteq R$. Thus $q \in P \cap R$. It follows that the APM structure built for the leaf will declare this point as lying inside $P \cap R$, and hence the overall algorithm will correctly declare that q lies in P .

Finally, we consider the case when $q \notin P$. To complete the proof, we need to show that if the distance of q from the boundary of P is greater than ε , then q is declared as lying outside P . Let p denote the point of intersection of the ray Oq with ∂P , let h denote a hyperplane supporting P at p , and let v denote the outward normal to h . Recall by part (i) of the claim that $\text{shadow}(E) \subseteq R$. It follows that h is a supporting hyperplane of $P \cap R$ at p . By part (ii) of the claim, $\text{width}_v(R) \leq c_2 d \delta$. It follows that $\text{width}_v(P \cap R) \leq c_2 d \delta$. Recall that the APM data structure for the leaf is built using approximation parameter $c_3 \varepsilon / \delta$ for some constant c_3 . By definition of APM query (in the standard, direction-sensitive sense), the absolute error allowed in direction v is at most $(c_3 \varepsilon / \delta) \cdot \text{width}_v(P \cap R) \leq (c_3 \varepsilon / \delta)(c_2 d \delta)$. By choosing c_3 sufficiently small we can ensure that this error is at most $\varepsilon \gamma$. To make this more precise, let h' denote the hyperplane parallel to h (outside P), and at distance $\varepsilon \gamma$ from it. Consider the halfspace bounded by h' and containing P . By definition of APM query, if q is not contained in this halfspace, then q would be declared as lying outside $P \cap R$, and the overall algorithm would declare q as lying outside P . Let p' denote the point of intersection of the ray Oq with h' . By Lemma 2.1(c), $\|pp'\| \leq (\varepsilon \gamma) / \gamma = \varepsilon$. Thus, if the distance of q from ∂P is greater than ε , then q cannot lie on segment pp' and q is correctly declared as lying outside P . This completes the proof of correctness. \square

We are now ready to prove Theorem 1.5

Proof. Our proof is based on a constant number of applications of the method presented in this section. It suffices to show that there is a data structure with space and query time as in the theorem and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(1/2+\beta')(d-1)})$, where $\beta' = \alpha/(d-1)$.

We initialize the bootstrapping process by the data structure described in the beginning of this section, which has $\beta = 1$. Recall that applying the method once changes the value of β to $\beta/(1 + \beta)$. It is easy to show that after i applications, the value of β will fall to $1/(i + 1)$. Thus, after $O(1/\alpha)$ applications, we will obtain a data structure with the desired preprocessing time. \square

6 Reductions

In this section, we show how the remaining problems reduce to polytope membership. We start with a useful variation of approximate nearest neighbor searching.

The input for an approximate nearest neighbor searching data structure is a set S of data points and an approximation parameter ε . Given a constant $\sigma > 0$, σ -well-separated approximate nearest neighbor searching is defined as follows. Let Q_S and Q_q be two hypercubes of side length r and at distance at least σr from each other. In the σ -well-separated version we have the data points S inside Q_S and the query points inside Q_q . Data structures for the well-separated version are much more efficient than for the unrestricted version. The following reduction from well-separated approximate nearest neighbor searching to approximate polytope membership is presented in [5, Lemma 9.2 of the journal version].

Lemma 6.1. *Let $0 < \varepsilon \leq 1/2$ be a real parameter, $\sigma > 0$ be a constant, and S be a set of n points in \mathbb{R}^d . Given a data structure for approximate polytope membership in d -dimensional space with query time $t_d(\varepsilon)$, storage $s_d(\varepsilon)$, and preprocessing time $O(n \log \frac{1}{\varepsilon} + b_d(\varepsilon))$ it is possible to preprocess S into a σ -well-separated ANN data structure with*

$$\text{Query time: } O\left(t_{d+1}(\varepsilon) \cdot \log \frac{1}{\varepsilon}\right) \quad \text{Space: } O(s_{d+1}(\varepsilon)) \quad \text{Preprocessing: } O\left(n \log \frac{1}{\varepsilon} + b_{d+1}(\varepsilon)\right).$$

Combining the previous reduction with Theorem 1.5 we have:

Lemma 6.2. *Given a set S of n points in \mathbb{R}^d , an approximation parameter $\varepsilon > 0$, and a constant $\sigma > 0$, there is a data structure that can answer σ -well-separated Euclidean ε -approximate nearest neighbor queries with*

$$\text{Query time: } O\left(\log^2 \frac{1}{\varepsilon}\right) \quad \text{Space: } O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{2}}\right) \quad \text{Preprocessing: } O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{d}{2} + \alpha}\right).$$

Next, we prove Theorem 1.3 using a reduction to well-separated approximate nearest neighbor searching that is based on [4, Theorem 3.2].

Proof. Let b denote the exact BCP distance. We obtain a constant approximation $b \leq a < 2b$ of the BCP distance in $O(n)$ expected time by running the randomized algorithm from [26]. Then, we build a grid with cells of diameter $a/4$ and partition the red points accordingly. Note that since $a/4 < b/2$, the BCP pair cannot be in the same grid cell, nor in two adjacent cells. The strategy of the algorithm is to partition the red points among the grid cells and to perform a constant number of well-separated approximate nearest neighbor queries for each blue point, returning the closest red-blue pair found. More precisely, for each blue point q , we perform an approximate nearest neighbor query among the grid cells Q_S that intersect the set theoretic difference of two balls of radii a and $a/2$ centered around q . These are the only grid cells that may contain the closest red point and, by a simple packing argument, the number of grid cells Q_S is constant. Since the grid cell Q_q that contains q cannot be adjacent to Q_S , it follows that the separation σ is at least 1.

To answer the queries efficiently, we separate the grid cells onto two types. If the number of red points in the cell is greater than $1/\varepsilon^{d/4}$, we say the cell is *heavy*, and otherwise we say the cell is *light*. Clearly, the number of heavy cells is $O(n \cdot \varepsilon^{d/4})$. We build well-separated approximate nearest-neighbor data structures for the heavy cells. Using Lemma 6.2, the total preprocessing time is $O(n/\varepsilon^{d/4 + \alpha})$. For each light cell, we simply store the red points it contains and answer nearest neighbor queries by brute force in $O(1/\varepsilon^{d/4})$ time. Therefore, the total time spent answering queries is $O(n/\varepsilon^{d/4})$. \square

An approximation to the Euclidean minimum spanning tree and minimum bottleneck tree can be computed by solving multiple BCP instances such that the sum of the number of points in all instances is $O(n \log n)$ [4, Theorem 4.1]. Applying this reduction together with Theorem 1.3, we prove Theorem 1.4.

The following reduction from ANN to APM is presented in [5, Lemma 9.3 of the journal version]. For the preprocessing time see [10, Lemma 8.3].

Lemma 6.3. *Let $0 < \varepsilon \leq 1/2$ be a real parameter and S be a set of n points in \mathbb{R}^d . Given a data structure for approximate polytope membership in d -dimensional space with query time at most $t_d(\varepsilon)$ and storage $s_d(\varepsilon)$, and preprocessing time $O(n \log \frac{1}{\varepsilon} + b_d(\varepsilon))$ it is possible to preprocess S into an ANN data structure with*

$$\begin{aligned} \text{Query time: } & O\left(\log n + t_{d+1}(\varepsilon) \cdot \log \frac{1}{\varepsilon}\right) & \text{Space: } & O\left(n \log \frac{1}{\varepsilon} + n \frac{s_{d+1}(\varepsilon)}{t_{d+1}(\varepsilon)}\right) \\ \text{Preprocessing: } & O\left(n \log n \log \frac{1}{\varepsilon} + n \frac{b_{d+1}(\varepsilon)}{t_{d+1}(\varepsilon)}\right). \end{aligned}$$

Applying this reduction with the data structure from Theorem 1.5 and setting $t_{d+1}(\varepsilon) = 1/(m \cdot \varepsilon^{d/2})$ for $\log \frac{1}{\varepsilon} \leq m \leq 1/(\varepsilon^{d/2} \log \frac{1}{\varepsilon})$, we obtain Theorem 1.7.

Next, we show how to obtain a data structure for approximate directional width queries (Theorem 1.6) using the data structure for approximate polytope membership from Theorem 1.5. The proof uses standard duality and binary search techniques.

Proof. Given a polytope P (defined as the intersection of n halfspaces) that contains the origin O , we define a ray-shooting query (from the origin) as follows. Let v be a query direction and let r denote the ray emanating from O in direction v . The result of the query $q(P, v)$ is the length of $r \cap P$. In the ε -approximate version, any answer between $q(P, v)$ and $(1 + \varepsilon)q(P, v)$ is acceptable.

If we place the origin O in the center of the John ellipsoid of P , we have $q(P, -v) = \Theta(q(P, v))$ for all v . Thus, a constant approximation of $q(P, v)$ can be obtained by replacing P by its circumscribing John ellipsoid. We can then refine the approximation using binary search and approximate polytope membership queries. (To see this, consider the point $p \in \partial P$ that is hit by the ray, and let h be any supporting hyperplane at p . Consider the slab containing P that is bounded by this hyperplane and the parallel hyperplane on the opposite side of P . By properties of the John ellipsoid, the origin lies within a central region of the slab. It follows from basic geometry that if we expand the slab by ε times its width, the ratio between ray distances to the expanded slab boundary and the original slab boundary is $1 + O(\varepsilon)$. An ε -APM query with respect to P along this ray will achieve an approximation error that is no greater.) By a suitable adjustment to the constant factor, we can obtain an ε -approximation to $q(P, v)$ after $O(\log \frac{1}{\varepsilon})$ membership queries.

The polar body P^* (defined as the convex hull of n points) of P has the property that $\text{width}_v(P^*) = 1/q(P, v) + 1/q(P, -v)$. Therefore, we can ε -approximate the width of a set of points P^* using $O(\log \frac{1}{\varepsilon})$ approximate polytope membership queries on P and Theorem 1.6 follows. \square

Agarwal, Matoušek, and Suri [3] showed that the diameter of a point set S can be ε -approximated by computing the maximum width of S among $O(1/\varepsilon^{(d-1)/2})$ directions. Therefore, Theorem 1.2 follows immediately from Theorem 1.6.

References

- [1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004.
- [2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*. MSRI Publications, 2005.
- [3] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1(4):189–201, 1992.

- [4] S. Arya and T. M. Chan. Better ε -dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε -kernels. In *Proc. 30th Annu. Sympos. Comput. Geom.*, pages 416–425, 2014.
- [5] S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd Annu. ACM Sympos. Theory Comput.*, pages 579–586, 2011. (Full version available from <http://arxiv.org/abs/1604.01183>, to appear on SIAM J. Computing).
- [6] S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal area-sensitive bounds for polytope approximation. In *Proc. 28th Annu. Sympos. Comput. Geom.*, pages 363–372, 2012.
- [7] S. Arya, G. D. da Fonseca, and D. M. Mount. On the combinatorial complexity of approximating polytopes. In *Proc. 32nd Internat. Sympos. Comput. Geom.*, pages 11:1–11:15, 2016. (Full version <http://arxiv.org/abs/1604.01175>, to appear on Discrete Comput. Geom.).
- [8] S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal approximate polytope membership. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 270–288, 2017. (Also available as <http://arxiv.org/abs/1612.01696>).
- [9] S. Arya, T. Malamatos, and D. M. Mount. The effect of corners on the complexity of approximate range searching. *Discrete Comput. Geom.*, 41:398–443, 2009.
- [10] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57:1–54, 2009.
- [11] S. Arya and D. M. Mount. A fast and simple algorithm for computing approximate Euclidean minimum spanning trees. In *Proc. 27th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 1220–1233, 2016.
- [12] S. Arya, D. M. Mount, and J. Xia. Tight lower bounds for halfspace range searching. *Discrete Comput. Geom.*, 47:711–730, 2012.
- [13] I. Bárány. Intrinsic volumes and f -vectors of random polytopes. *Math. Ann.*, 285:671–699, 1989.
- [14] I. Bárány. The technique of M-regions and cap-coverings: A survey. *Rend. Circ. Mat. Palermo*, 65:21–38, 2000.
- [15] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001.
- [16] J. L. Bentley, M. G. Faust, and F. P. Preparata. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, 1982.
- [17] H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.
- [18] E. M. Bronshteyn and L. D. Ivanov. The approximation of convex sets by polyhedra. *Siberian Math. J.*, 16:852–853, 1976.
- [19] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006.
- [20] T. M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 26:1–15, 2017.
- [21] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21:579–597, 1996.
- [22] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

- [23] K. Dutta, A. Ghosh, B. Jartoux, and N. H. Mustafa. Shallow packings, semialgebraic set systems, Macbeath regions and polynomial partitioning. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 38:1–15, 2017.
- [24] G. Ewald, D. G. Larman, and C. A. Rogers. The directions of the line segments and of the r -dimensional balls on the boundary of a convex body in Euclidean space. *Mathematika*, 17:1–20, 1970.
- [25] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204. Interscience Publishers, Inc., New York, 1948.
- [26] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, 1995.
- [27] A. M. Macbeath. A theorem on non-homogeneous lattices. *Ann. of Math.*, 56:269–293, 1952.
- [28] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.
- [29] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114–127, 1984.
- [30] N. H. Mustafa and S. Ray. Near-optimal generalisations of a theorem of Macbeath. In *Proc. 31st Internat. Sympos. on Theoret. Aspects of Comp. Sci.*, pages 578–589, 2014.